HW 4

Problem 1.

In this exercise, we examine in detail how an instruction is executed in a single-cycle datapath.

Problems in this exercise refer to a clock cycle in which the processor fetches the following

instruction word: 0x00c6ba23.

1.1 What are the values of the ALU control unit’s inputs for this instruction?

1.2 What is the new PC address after this instruction is executed? Highlight the path

through which this value is determined.

1.3 For each mux, show the values of its inputs and outputs during the execution of this

instruction. List values that are register outputs at Reg [xn].

1.4 What are the input values for the ALU and the two add units?

1.5 What are the values of all inputs for the register’s unit?

Answers:

1.1

0x00c6ba23 = 110001101011101000100011(binary)

ALUop instrcutrion are 00.

The value of ALU Control line are 0010.

1.2 the new pc address after the instruction would be pc + 4

1.3

There are three Mux: ALUsrc, PCsrc, MemtoReg

ALUsrc:

Control input would be 1

Input: Reg[x12] and 0x0000000000000014

The output would be: 0x0000000000000014

PCsrc

Control input 0

Inputs: PC + 4 and

Output PC + 4

MemotoReg

Control input : 0

Input: Reg[x13] + 0x14 and undefine.

Output: undefined.

1.4

Input value: Reg[x13] (RS1) and 0x0000000000000014

PC Adder inupts: PC and 4

Branch adder inputs: PC and 0x000000000000014 (shift by 1 from 0x 000000000028)

1.5 :

Read Register 1: 01101 identify the register 13

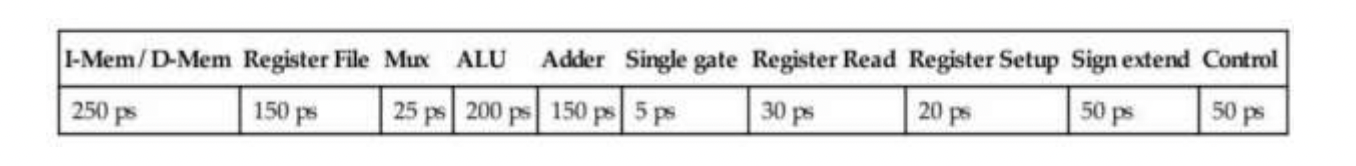
Read register 2: 01100, which is the register 11

Write register : undefine bits,

Reg write control signal disabled.

2. Problems in this exercise assume that the logic blocks used to implement a processor’s datapath

have the following latencies:



“Register read” is the time needed after the rising clock edge for the new register value to appear on

the output. This value applies to the PC only. “Register setup” is the amount of time a register’s data

input must be stable before the rising edge of the clock. This value applies to both the PC and

Register File.

2.1 What is the latency of an R-type instruction (i.e., how long must the clock period be to

ensure that this instruction works correctly)?

2.2 What is the latency of ld? (Check your answer carefully. Many students place extra muxes

on the critical path.)

2.3 What is the latency of sd? (Check your answer carefully. Many students place extra muxes

on the critical path.)

2.4 What is the latency of beq?

2.5 What is the latency of an I-type instruction?

2.6 What is the minimum clock period for this CPU?

Answers :

2.1 the latency of R type instruction would be

Total latency = pc register + instruction memory + read register + mux + ALU + MUX \* 2 + register setup= 30 + 250 + 150 + 25 + 200 + 50 + 20 = 700ps

2.2

Total latency of ld would be :

= register read + instruction Memory + register file + MUX \* 2 + ALU + D-Mem + register setup

= 30 + 250 + 150 + 25 + 300 + 350 + 25 + 20

= 950

2.3

Total latency of sd = register read + i\_mem + read register + Mux + ALU + D-M = 30 + 250 + 150 + 25 + 200 + 250 + 905

2.4

Latency pf beq = register read + i-M + read register + MUX + ALU + single gate + Mux + setup =

= 30 + 250 + 150 + 25 + 200 + 5 + 25 + 20 = 705

2.5

Latency of I-type instruction would be pc register + instruction memory + read register + mux + ALU + MUX \* 2 + register setup= 30 + 250 + 150 + 25 + 200 + 50 + 20 = 700ps

2.6

Minimum clock period would be 950 ps

3. (a) Suppose you could build a CPU where the clock cycle time was different for each instruction.

3.a1 What would the speedup of this new CPU be over the CPU presented in Figure 4.21

(in RISC-V text) given the instruction mix below? (assuming instruction latencies from the

problem 2)

Table

Description automatically generated

Cycle = 700 ps \* 0.52 + 950 \* 0.25 + 905 \* 0.12 = 785.6 ps

Speedup = old / new = 950 / 785.6 = 1.21

3 (b) Consider the addition of a multiplier to the CPU shown in Figure 4.21. This addition will add

300 ps to the latency of the ALU, but will reduce the number of instructions by 5% (because there

will no longer be a need to emulate the multiply instruction).

3.b1 What is the clock cycle time with and without this improvement?

3.b2 What is the speedup achieved by adding this improvement?

3.b3 What is the slowest the new ALU can be and still result in improved performance?

3.b1

With multiplier improvement would be 950 ps

Without improvement would be 1250 ps

3.b2

Speedup = old / new = 950 / (0.95 \* 1250) = 0.8

3.b3

Reduce 5 percent of the instructions in the CPU , it would increase 50ps of the cycle time,

still insufficient to cover up 300ps , therefore, ALU stay the same.

3 (c) When processor designers consider a possible improvement to the processor datapath, the

decision usually depends on the cost/performance trade-off. In the following three problems,

assume that we are beginning with the datapath from Figure 4.21, the latencies from Problem 2

in this assignment, and the following costs:

Suppose doubling the number of general-purpose registers from 32 to 64 would reduce the number

of ld and sd instruction by 12%, but increase the latency of the register file from 150 ps to 160

ps and double the cost from 200 to 400. (Use the instruction mix [from 3(a) above] and ignore the

other effects on the ISA)

Table

Description automatically generated

3.c1 What is the speedup achieved by adding this improvement?

3.c2 Compare the change in performance to the change in cost.

3.c3 Given the cost/performance ratios you just calculated, describe a situation where it makes

sense to add more registers and describe a situation where it doesn’t make sense to add more

registers.

3.c1

number of instruction reduced, would be 100 - 12 \* (25 + 11) = 100 % - 4.3% = 95.7%

since the register file from 150. To 160 then we have

95.7% \* 960 = 918.7

Speedup = 960/918.7 = 1.04

3.c2

In crease cost would be 4196 / 3995 = 1.05 , which is 5% increased

3.c3

Suppose added 32 register, then 12% Id sd would be reduced, 200 incread cost, which is 4.4 % st and 3 percent increased improvement of performance.

4. ld is the instruction with the longest latency on the CPU from Section 4.4 (in RISC-V text).

If we modified ld and sd so that there was no offset (i.e., the address to be loaded from/stored

to must be calculated and placed in rs1 before calling ld/sd), then no instruction would use

both the ALU and Data memory. This would allow us to reduce the clock cycle time. However,

it would also increase the number of instructions, because many ld and sd instructions would

need to be replaced with ld/add or sd/add combinations.

4.1 What would the new clock cycle time be?

4.2 Would a program with the instruction mix presented in Problem 2 run faster or slower

on this new CPU? By how much? (For simplicity, assume every ld and sd instruction is replaced

with a sequence of two instructions.)

4.3 What is the primary factor that influences whether a program will run faster or slower

on the new CPU?

4.4 Do you consider the original CPU (as shown in Figure 4.21 of RISC-V text) a better

overall design; or do you consider the new CPU a better overall design? Why

4.1

The new clock time would be 750 ps

The new sd instruction would. Be 705 ps

4.2

new cycle time = 750ps

would be multiplied with 1.36n to give 1020n ps. speedup = 950/1020 = 0.93 or a reduction in speed by 7%

4.3

The primary factor that influences whether a program wil run faster or slower on the new CPu would be the frequency of load/ store instructions. IF Frequency is a frequency to which a carrier frequency is shifted as an intermediate step in transmission or reception. And it is the factor that influence the improvement in cycle time.

4.4

It depends on the frequency of load/stored instruction. The improvement to cycle time is 22% by eliminating the ALU op for ld/sd. However, if ld/sd occur 22% of the instructions then it is not better for the CPU.

5. (a) Examine the difficulty of adding a proposed lwi.d rd, rs1, rs2 (“Load With

Increment”) instruction to RISC-V. Interpretation: Reg[rd]=Mem[Reg[rs1]+Reg[rs2]]

5.a1 Which new functional blocks (if any) do we need for this instruction?

The ALU could be used to add the 2 registers rs1 and rs2, the shifter could be used to shift the offset in register rs2, so no new hardware is necessary

5.a2 Which existing functional blocks (if any) require modification?

Instruction memory, one register read ports, the path that passed the immediate to the ALU, and the register write port.

5.a3 Which new data paths (if any) do we need for this instruction?

We need to extend the existing ALU to also do shifts (SLL, to extend the offset to

32bit value).

6. In this exercise, we examine how pipelining affects the clock cycle time of the processor.

Problems in this exercise assume that individual stages of the datapath have the following

latencies:

Also, assume that instructions executed by the processor are broken down as follows:

6.1 What is the clock cycle time in a pipelined and non-pipelined processor?

6.2 What is the total latency of an ld instruction in a pipelined and non-pipelined processor?

6.3 If we can split one stage of the pipelined datapath into two new stages, each with half the

latency of the original stage, which stage would you split and what is the new clock cycle time of

the processor?

6.4 Assuming there are no stalls or hazards, what is the utilization of the data memory?

6.5 Assuming there are no stalls or hazards, what is the utilization of the write-register port of

the “Registers” unit?

6.1

Piplined : 350 ps

Nonpipline: 1250 ps

6.2

Both piplined and nonpiplined are 1250 ps

6.3

ID stage : 350ps

New pipelined stage: 300ps

6.4

Load+Store = 35%

6.5

65% - only ALU and load

7. What is the minimum number of cycles needed to completely execute n instructions on a CPU

with a k stage pipeline? Justify your formula.

in a k stage pipeline:

the first instruction reached the end of the pipeline in k cycles.

Then, the n-1 instructions execute at 1 cycle/instruction over the next n-1 cycles.

So n instructions take k+n-1 cycles to execute in a k stage pipeline.

8. (a) Assume that x11 is initialized to 11 and x12 is initialized to 22. Suppose you executed the

code below on a version of the pipeline from Section 4.5 that does not handle data hazards (i.e.,

the programmer is responsible for addressing data hazards by inserting NOP instructions where

necessary). What would the final values of registers x13 and x14 be?

addix11, x12, 5

addx13, x11, x12

addix14, x11, 15

x13 = 33,

x14 = 26

(b) Assume that x11 is initialized to 11 and x12 is initialized to 22. Suppose you executed the

code below on a version of the pipeline from Section 4.5 that does not handle data hazards (i.e.,

the programmer is responsible for addressing data hazards by inserting NOP instructions where

necessary).

What would the final values of register x15 be? Assume the register file is written at the beginning

of the cycle and read at the end of a cycle. Therefore, an ID stage will return the results of a WB

state occurring during the same cycle. See Section 4.7 and Figure 4.51 for details.

addix11, x12, 5

addx13, x11, x12

addix14, x11, 15

addx15, x11, x11

the x13 = 33

x11 = 26

x15 = 54

(c) Add NOP instructions to the code below so that it will run correctly on a pipeline that does not

handle data hazards.

addix11, x12, 5

nop

nop

addx13, x11, x12

addix14, x11, 15

nop

addx15, x13, x12